Email Embeddings for Phishing Detection

Luis Felipe Gutiérrez[1], Faranak Abri[1], Miriam Armstrong[2], Akbar Siami Namin[1], and Keith S. Jones[2]
[1]Department of Computer Science, [2]Department of Psychological Sciences
Texas Tech University
{*Luis.Gutierrez-Espinoza, faranak.abri, miriam.armstrong, akbar.namin, keith.s.jones*}*@ttu.edu*

2020 IEEE International Conference on Big Data (Big Data) | 978-1-7281-6251-5/20/$31.00 ©2020 IEEE | DOI: 10.1109/BigData50022.2020.9377821

***Abstract*—The problem of detecting phishing emails through machine learning techniques has been discussed extensively in the literature. Conventional and state-of-the-art machine learning algorithms have demonstrated the possibility of building classifiers with high accuracy. The existing research studies treat phishing and genuine emails through general indicators and thus it is not exactly clear what phishing features are contributing to variations of the classifiers. In this paper, we crafted a set of phishing and legitimate emails with similar indicators in order to investigate whether these cues are captured or disregarded by email embeddings, i.e., vectorizations. We then fed machine learning classifiers with the carefully crafted emails to find out about the performance of email embeddings developed. Our results show that using these indicators, email embeddings techniques is effective for classifying emails as phishing or legitimate.**

***Index Terms*—Natural Language Processing, Phishing Emails, Email Embeddings**

I. Introduction

Autonomous devices are the integral part of cyber-Physical Systems (CPS) [1]. These devices are powered by elegant software applications and utilities to protect against adversarial attacks. Information theoretical solutions along with AI-enabled autonomous agents are capable of protecting these critical components from cyber attacks. However, in addition to these hardware and software devices, there is another autonomous entity in CPS that is harder to harden their security defense systems, i.e., humans.

It is known that humans are the weakest link in the information security chain. Adversarial attacks often exploit this weakness through sending malicious links to individuals who are operating critical infrastructure with the hope that the operators visit malicious links and Websites. These types of phishing attacks can be in the form of emails, adware, or even malicious fake Websites [2].

In spite of advancement in security technologies and controls, cyber security attacks through phishing still remain the number one choice of attackers due to their higher success rates. Even though many antivirus software and blocking strategies are able to detect spamicity of emails, these malicious emails are able to get the attention of the target receivers, pass through the spam detection tools, and thus being activated.

Machine learning-based approaches have revolutionized the detection mechanism of spam and phishing emails. However, the goodness and accuracy of the detection power of these learning-based algorithms heavily rely on the type of historical data, as known as training data. Existing studies confirm that phishing emails are distinguishable from genuine ones based on features. However, it is unclear whether the historical data are rich enough to train the classifiers properly. More specifically, an important question is whether the machine learning detection-based models are content-aware or they are content agnostic.

To address this grand problem, this paper studies the performance of email embeddings (i.e., email vectorization) to detect phishing emails. We created 12 genuine and 12 phishing emails. The systematically crafted emails are then vectorized and fed into well-known machine learning classifiers. The results demonstrate that, even though the contents of both phishing and legitimate emails are similar, the email embeddings technique is able to distinguish between phishing and genuine emails. To capture the content of the emails into account, we then embed emails using doc2vec, a vectorization approach to capture the semantic of a given text.

In our previous work [2], [3], we studied the usage of linguistic features in the context of fake reviews. We observed that it is possible to detect fake reviews using linguistic features. In this paper, we are interested in exploring whether it is possible to detect phishing emails when the contents of both phishing and legitimate emails are somewhat similar. The key contributions of this paper are as follows:

- We introduce a carefully and systematically crafted set of phishing and legitimate emails to support this line of research.
- The performance of email embeddings techniques is presented through a number of machine learning classifiers.
- The results show that even in the presence of similar contents, the email embeddings are able to distinguish legitimate emails from the phishing ones.

This paper is organized as follows: Section II reviews the related work in this line of research. In Section III, a brief technical background of the machine learning models and embeddings is presented. Section IV presents the experimental setup and procedure. The results of the study are presented in Section V. Section VI concludes the paper and highlights future research directions.

II. Related work

Machine learning techniques are broadly used for spam detection. The most important step for developing such a detection framework is extracting features that are fed into

classifiers. A very common method used for this step is Bag of Words (BOW), which utilizes the occurrence frequency of the words or group of words. Although this method is computationally fast and easy to implement, it has several limitations such as ignoring semantics, word order, and a huge number of features that imposes the curse of dimensionality on machine learning classifiers [4].

Term frequency-inverse document frequency (tf-idf) is another method for feature extraction that is commonly used in search engines and is useful for spam detection. It considers the frequency of only the key words by checking the occurrence frequency of a word in a text or document and comparing it to its occurrence in the whole document or corpus. Therefore, words that are repeated in small sections have higher scores compared to commonly used words in the whole content.

BOW or tf-idf are less efficient methods for detecting spams, such as phishing emails, which have more specific content properties such as a URL link to a malicious source. Given that sometimes it is desirable to pick out phishing emails especially from spam emails, more robust features are needed. Doc2Vec [5] is a document embedding technique in which similar documents have similar encodings semantically. It is an unsupervised neural network, which predicts the words in a document.

Duzi et al. [6] proposed an ensemble method that detects spam emails by combining the classifying results obtained from two feature sets. These feature sets extracted using Doc2Vec and tf-idf methods. They compared different classifiers with these feature sets on two datasets: $33,702$ emails from the Enron dataset [7] and $2,314$ emails from the Ling spam corpus. Using support vector machines, they achieved 98.27% for accuracy and 98.97% for f-score on the Ling spam dataset and 96.16% for accuracy and 96.07% for f-score on the Enron dataset, respectively.

Akinyelu and Adewumi [8] evaluated a random forest (RF) model for classifying phishing emails from legitimate ones. Using two datasets [9] for legitimate emails and [10] for phishing emails, a new dataset with a total of $2,000$ emails was made. They extracted 15 features such as "URLs Containing IP Address" and "Presence of Javascript" from samples. Using 10 fold cross validation by a RF classifier, the best accuracy achieved with 99.7% along with 99.47%, 97.5%, 98.45% for precision, recall and f-score, respectively. They also examined their model on datasets with different sizes and showed that the performance of the classifier increases using more email samples.

Unnithan et al. [11] examined several classifiers for phishing email detection. They used the dataset shared by the "IWSPA-AP 2018" workshop, including train and test subsets for emails with headers and emails without headers separately. The subset of no-header emails contains $4,300$ samples for the test data and $5,700$ emails as train data, including $5,088$ legitimate and 612 phishing emails. Two sets of features were generated using the Term frequency-inverse document frequency (tf-idf) and Doc2Vec techniques. They trained seven different models including: Decision Tree (DT), Naive Bayes (NB), Ada-boost, Logistic Regression (LR), K-nearest neighbour (KNN), Support vector machine (SVM), and Random Forest (RF). They applied the models to both subsets of the data (emails with and without headers) using both feature sets. Using the SVM classifier with Doc2Vec feature set on both datasets, they achieved the best results including: $3,825$ true positives (TP), 0 true negative (TN), 475 false positive (FP), and 0 false negative (FN) on emails without header.

III. Models and Algorithms

A. Feature Extraction

1) Doc2Vec: Doc2Vec is a technique to generate document embeddings through a neural network-based approach [5], similar to what Word2Vec [12] achieves with words. Figure 1 shows the Doc2Vec model adapted from [5]. As it is observable, the inclusion of a *paragraph id* in the training phase is a difference with respect to the original Word2Vec model. Note that although in [5] the term used is *paragraph id*, normally, this method is used to embed full documents. Given that all words in a specific document are trained using the id assigned to the document, the resulting vector (i.e., embedding) for the document encodes meaning according to the words it contains. Furthermore, as both word and document embeddings are generated during the same training procedure, both words and documents are embedded in the same semantic vector space.

Hence, the resulting vector space has the capability to encode semantic similarities, i.e., vectors that are close together tend to share semantic properties, as Doc2Vec works under the Distributional Hypothesis of linguistics [12] (i.e., words that are used in similar contexts tend to hold similar meaning).

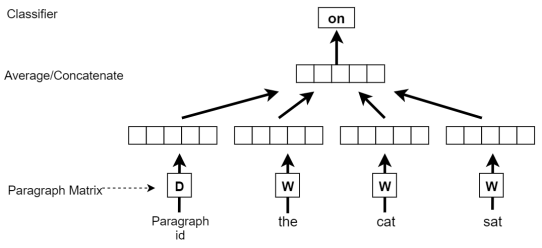

Fig. 1. Doc2Vec model (adapted from [5]).

Doc2Vec generates dense feature vectors, in contrast to the sparse representations produced by frequency-based techniques (e.g., BOW). Because of this, there exists a trade-off between sparsity of the features and interpretability, as the features extracted by Doc2Vec are considered opaque, since there is no direct interpretation for the values in each dimension.

B. Dimensionality Reduction

1) Principal Component Analysis: Principal Component Analysis (PCA) is a technique that is commonly used in dimensionality reduction [13]. The dimensionality reduction is performed by considering the m principal components in

the dataset, while retaining as much variance of the original dataset as possible. However, there are three important assumptions of the given data when PCA is utilized [13]:

1) Linearity of the data should be present,
2) Large variances denote an important structure, and
3) The principal components are orthogonal.

In addition, when PCA is used for visualization purposes, the value for m (i.e., the number of principal components) is 2 or 3. In this work, we set $m = 2$ to visualize the data and also to explore the results of classification using the 2-D projection of the document embeddings.

2) Kernel Principal Component Analysis: In some cases, the linear nature of the data that PCA takes as an assumption might not hold, and a non-linear structure could explain the underlying phenomena more accurately. Using this idea, PCA can still be applied to such datasets using a non-linear mapping that takes the samples in the original space, often called input space, to a higher dimensional space, also called feature space, and then performing PCA in the resulting feature space [14]. The mapping function is often referred to as *Kernel*, and the full algorithm as *Kernel* PCA. Like the case of SVMs, several types of kernels can be utilized to model different patterns in the data. In our work, we used the Gaussian or RBF kernel.

C. Classifiers

Brief descriptions of the classifiers that we experimentally compared are as follows:

1) Random Forest: The random forest classifier, also known as ensemble of decision trees, combines the prediction of several weak classifiers (i.e. decision trees) and produces a final classification through voting. The main hyperparameter of this model is the number of weak classifiers to combine. Due to the fact that decision trees are weak classifiers, their corresponding hyperparameters are also present in this model.

2) Support Vector Machine: Support Vector Machines (SVMs) determine a hyperplane that maximizes the margin between itself and the nearest samples of each data class. Such nearest samples are known as support vectors. This construction can be performed in the original input space, or a feature space that is generated by applying a kernel function to each pair of samples, which allows the decision boundary to be non-linear. In our work, we used the linear, Radial Basis Function (RBF), polynomial, and sigmoid kernels. An SVM converges towards non-linearly separable classes using the C hyperparameter, where C controls the number of samples that can be misclassified, and its optimal value is problem-dependent. Moreover, there exist additional hyperparameters for each kernel, such as the degree in the polynomial kernel, or the gamma term in the RBF kernel.

3) Logistic Regression: In binary classification, logistic regression assigns the probability that a sample, denoted by the feature vector x, belongs to a class C_1 as [15]:

$$P(C_1|x, \alpha) = \frac{1}{1 + \exp(-\alpha \cdot x)}$$

where α is the set of parameters of the model. The probability $P(C_2|x, \alpha)$ is calculated as $1 - P(C_1|x, \alpha)$. Usually, the learning of the α parameter is conducted through the optimization of the cross-entropy loss function [15].

4) Naive Bayes: The Naive Bayes classifier performs classification using the strong assumption that all features in a vector $X = (X_1, ..., X_n)$ are statistically independent with respect to the class C. This is,

$$P(X|C) = \prod_{i=1}^{n} P(X_i|C).$$

This model assigns a predicted class label $\hat{y}$ according to

$$\hat{y} = \arg\max_k P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

where C_k is the k-th class.

Because the features of our dataset are generated using Doc2Vec document embeddings, we used the Gaussian Naive Bayes in order to estimate the conditional probabilities with a dataset of continuous features. Given a class C_k, a variance σ_k^2, a mean μ_k (both σ_k^2 and μ_k are estimated from the samples), and an observed value $\hat{x}$ for the feature, the probability $P(x = \hat{x}|C_k)$ can be estimated using

$$P(x = \hat{x}|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(\hat{x} - \mu_k)}{2\sigma_k^2}\right).$$

Although this assumption is unnatural and it does not hold for most situations, the Naive Bayes classifier has been adopted successfully in tasks such as spam filtering, medical diagnosis, and text classification [16]

IV. Experimental Setup

A. Scripts and Libraries

We developed Python 3 scripts for running our experiments. We used the classifiers' implementation provided by the Scikit-learn library. To obtain the Doc2Vec document embeddings, we used the Gensim library [17]. For stemming, we used the Porter algorithm provided by the NLTK library [18].

B. Data Collection

Twenty-four email stimuli were created for an experiment with human subjects. The content of the emails was modeled off legitimate emails found in the authors' inboxes or online. For all emails, a sender address, subject line, email body, and URL were created. The body of all emails was 50-100 words long and contained a hyperlink. Because the email stimuli were created to later show to research participants, this method for creating emails and the characteristics of the email stimuli are similar to those used in other phishing experiments [19]–[21].

Twelve of the 24 email stimuli were legitimate emails, and the other 12 emails were phishing. All legitimate emails shared two characteristics. First, the URL in the legitimate emails went to a real and safe web address. All legitimate email URLs began with HTTPS. Second, the body of legitimate emails contained targeted messaging. All email stimuli were addressed to a fictional university student. Email stimuli with targeted messaging referenced the student by name, referenced their position as a student, or referenced the city in which the student attended university.

The 12 phishing emails were designed to mimic non-targeted phishing emails. The non-targeted phishing emails differed from the other types of emails in terms of both their URL and email body. The URLs of the non-targeted phishing emails did not contain HTTPS and had at least two additional characteristics of suspicious links [22]: five or more dots in the domain, over 75 characters long, contained an IP address, or contained misspellings. The email body contained no targeted messaging; non-targeted phishing emails were designed to be emails to could be sent to anyone.

C. Data Preprocessing

We used standard techniques for text preprocessing in order to clean the dataset. These are removal of 1) punctuation marks, 2) URLs, 3) email addresses, and 4) stopwords. We also used the Porter stemmer [23] to perform stemming for each remaining token.

D. Classification Metrics

Consider a confusion matrix with counts of samples classified as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). We report the results of our experiments using accuracy, F_1 measure, precision, and recall, which are defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN},$$

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

E. Hyperparameters Tuning

We performed several exhaustive grid search in order to find the best set of hyperparameters for SVM, Logistic Regression, and Random Forest. For SVM, we tried different values for the C constant, kernel type, degree of the polynomial in case of using polynomial kernel, and the gamma value in case of the Gaussian kernel. For logistic regression, we tried different values for the regularization parameter. For random forests, we tried different number of estimators, maximum tree depth, minimum number of samples per leaf, minimum number of samples per split, and criterion to choose one split over others.

F. Experiments Flowchart

Figure 2 shows the flowchart of the experiments carried out in our work. After the feature extraction step, there are three scenarios in which the classification task was applied: 1) directly using the 20-dimensional Doc2Vec email embeddings, 2) using the linear PCA 2-D projections of the embeddings, and 3) using the Kernel PCA 2-D projections of the embeddings. Whether it is using the full high-dimensional features or a 2-D projection of it, the next step is the hyperparameter tuning, in which the best hyperparameters for the classifiers are set. Once the hyperparameters are found, the classifiers are fit using them and report the classification metrics using 10-fold cross-validation.

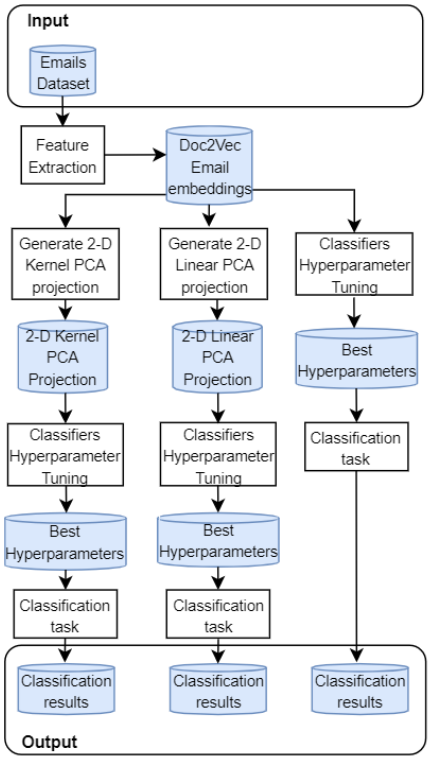

Fig. 2. Flowchart of the experiments conducted in our work.

V. Results and Discussion

A. Classification

We report the averages of accuracy and F_1 score obtained after running the experiments using 10-fold cross-validation. Table I shows the results for the classifiers using the full 20-dimensional space provided by the document embeddings. SVM reports accuracy and F_1 score of 81.6% and 76.6%, respectively, which is significantly higher than that of the next classifier, the random forest.

TABLE I
Performance scores for the classifiers in the 20-dimensional Doc2Vec feature space.

Classifier	Accuracy	F_1 score	Precision	Recall
SVM	**0.816**	**0.766**	**0.750**	**0.750**
Logistic Regression	0.616	0.566	0.600	0.700
Random Forest	0.583	0.483	0.533	0.600
Naive Bayes	0.600	0.553	0.616	**0.750**

Table II shows the results for the classifiers trained using only the first two components projections of linear PCA. In general, these results are consistently higher than those of in Table I. This is an interesting finding as it suggests that the variance present in the first two components of PCA is not only enough to maintain the performance of the classifiers, but it also increases the performance. Random forest reports the highest accuracy and F_1 score with 91.6% and 90.0%, respectively. However the margin between the performance of random forest and the rest of the classifiers is also reduced, the SVM and Naive Bayes classifiers having similar values performance-wise.

TABLE II
PERFORMANCE SCORES FOR THE CLASSIFIERS IN THE 2-D LINEAR PCA SPACE.

Classifier	Accuracy	F_1 score	Precision	Recall
SVM	0.900	0.866	0.800	0.800
Logistic Regression	0.750	0.700	0.650	0.600
Random Forest	**0.916**	**0.900**	0.800	0.750
Naive Bayes	0.900	0.866	**0.850**	**0.900**

Table III shows the results for the classifiers trained using the first two components projections of the RBF Kernel PCA. Unlike the results for linear PCA, the performance of the classifiers is diminished with respect to the results in the original 20-dimensional space. In this case, the best results are reported by SVM, with an accuracy and F_1 score of 78.3% and 76.6%, respectively.

TABLE III
PERFORMANCE SCORES FOR THE CLASSIFIERS IN THE 2-D RBF KERNEL PCA SPACE.

Classifier	Accuracy	F_1 score	Precision	Recall
SVM	**0.783**	**0.766**	**0.800**	0.700
Logistic Regression	0.600	0.520	0.516	**0.800**
Random Forest	0.700	0.700	0.750	0.700
Naive Bayes	0.600	0.520	0.516	0.800

One interesting finding of these results is that the linear PCA representation of the document embeddings yields to better classification results. This is worth noting as the document embeddings often contain an underlying non-linear structure. Because Doc2Vec embeds the document vectors in a semantic vector space, this suggests that the content of the emails can be segmented according to the class through a linear transformation of the vectors.

B. Visualization of Email Embeddings

Figure 3 shows the 2-dimensional plot for the email embeddings using the first two linear PCA components where the x-axis represents the first principal and the y-axis holds the values for the second principal. Additionally, we plotted the decision boundary of a SVM with RBF kernel that was fitted using the 24 emails only for visualization purposes. It is visible that the distribution of document embeddings follows a pattern that is easily captured by the decision boundary of the SVM. In this projection, phishing emails are contained in a blob located near the center of the plot, with only two legitimate emails within or in the border of the decision boundary. This finding agrees with the notoriously high performance of the classifiers using the linear PCA 2-D projections. However, note that even though the projection is obtained as linear, the decision boundary is not linear, which also explains the high performance for non-linear classifiers.

Figure 4 shows the 2-dimensional plot using the first two RBF Kernel components. Like Figure 3, we also plotted the decision boundary of a SVM that was fitted using all the samples and a RBF kernel. However, unlike Figure 3, the document embeddings are not clearly segmented by class. Legitimate emails are grouped closely together with the exception of emails 9 and 2, which are the same emails that

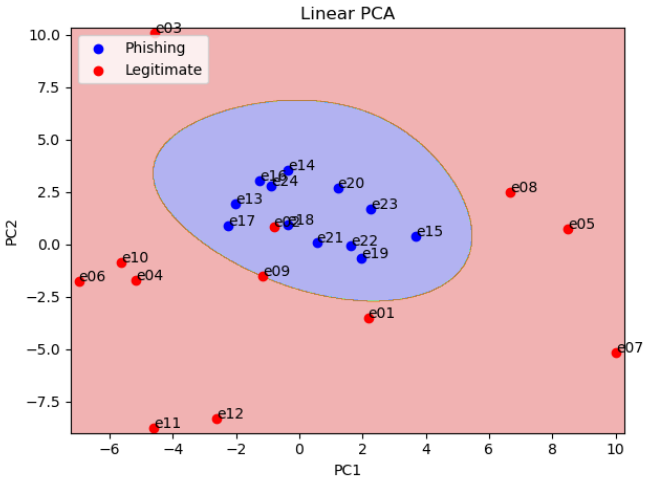

Fig. 3. SVM: decision boundary on the 2-D linear PCA projection.

lie within or near the wrong side of the decision boundary in Figure 3. Nevertheless, there are several phishing emails on the wrong side of the decision boundary. In this sense, this projection presents a less useful representation for class segmentation when compared to linear PCA, which can be seen in its classification results in Table III. The slight difference between the results in Table I and Table III might suggest that the document embeddings follow a similar configuration in the 20-dimensional feature space.

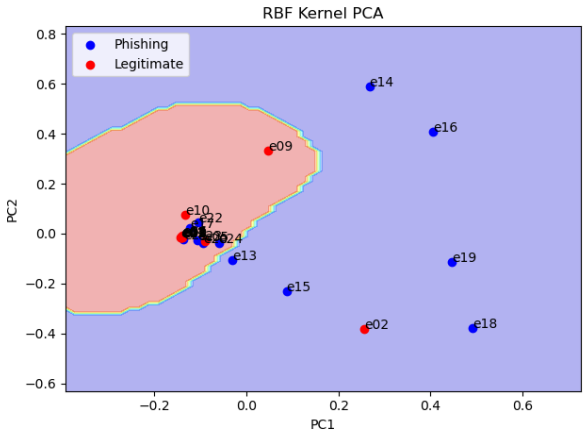

Fig. 4. SVM: decision boundary on the 2-D Kernel PCA projection.

C. Explained Variance of Linear PCA

Given that we obtained better classification results using the linear PCA projection, we performed an explained variance analysis for this PCA. Figure 5 shows the cumulative explained variance per number of components using PCA. The x-axis represents the number of components; whereas, the y-axis shows the cumulative explained variance ratio R_n, defined as $R_n = \sum_{i=1}^{n} r_i$, where n is the number of components and r_i is the portion of variance explained by the i-th component. In this plot, the 90% of the explained variance is reached using 12

components. The results in Table II are reported using the first two PCA components, which approximately comprise the 25% of the total variance. Hence, one surprising finding is that the 91.6% accuracy of the random forest classifier was achieved using only 25% of the original variance.

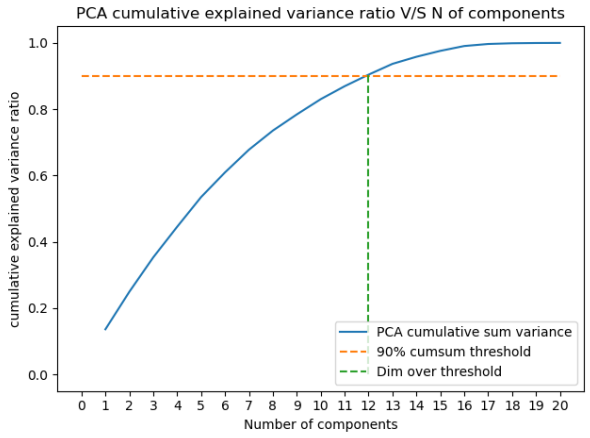

Fig. 5. Cumulative explained variance of linear PCA vs. # of components.

VI. Conclusion and Future Work

In this work, we generated document embeddings using Doc2Vec and performed binary classification using SVM, Logistic Regression, Random Forest, and Naive Bayes. We executed the experiments on the full 20-dimensional feature space generated by Doc2Vec, and additionally we calculated the 2-D projections of linear PCA and RBF Kernel in order to run the experiments in the low-dimensional projections.

In the original feature space, SVM reports the highest classification performance with an accuracy and F_1 score of 81.6% and 76.6%, respectively. The results using the 2-D linear PCA projections are higher than those of the 20-dimensional feature space, where the Random Forest reports an accuracy and F_1 score of 91.6% and 90.0%, respectively, using only 25% of the original variance. The results using the 2-D RBF Kernel PCA projections are slightly lower than those of the 20-dimensional feature space, where the SVM reports an accuracy and F_1 score of 78.3% and 76.6%, respectively. The highest results using the linear PCA projections suggests that the underlying structure of the Doc2Vec document embeddings is likely to be linear. The overall high classification results suggest that the semantic vector space in which the document vectors are is appropriate for this classification task. Moreover, the semantics of the emails' content are well suited for the class segmentation.

As future work, we will explore the use of features that permit an easier interpretation and provide a deeper insight into phishing and legitimate emails. There are some other intriguing approaches to address the phishing email detection problem. A possible approach is the use of evidence theory and fusion in formulating the problem [1] where a set of linguistic features and evidence can be used to decide pignistic probability of whether an email is phishing. It is also possible to model the phishing email detection through exploring some other machine learning techniques [24] or emerging deep/machine learning techniques such as reinforcement learning [22].

Acknowledgment

This research work is supported by National Science Foundation (NSF) under Grant No: 1723765.

References

[1] M. Chatterjee, A. S. Namin, and P. Datta, "Evidence fusion for malicious bot detection in IoT," in *IEEE Big Data*, 2018, pp. 4545–4548.
[2] F. Abri, L. Gutierrez-Espinoza, A. S. Namin, K. S. Jones, and D. R. W. Sears, "Linguistic features for detecting fake reviews," in *ICMLA*, 2020.
[3] L. Gutierrez-Espinoza, F. Abri, A. S. Namin, K. S. Jones, and D. R. W. Sears, "Ensemble learning for detecting fake reviews," in *IEEE COMPSAC*, 2020, pp. 1320–1325.
[4] "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, p. e01802, 2019.
[5] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, 2014, pp. 1188–1196. [Online]. Available: http://proceedings.mlr.press/v32/le14.html
[6] S. Douzi, F. AlShahwan, M. Lemoudden, and B. Ouahidi, "Hybrid email spam detection model using artificial intelligence," *International Journal of Machine Learning and Computing*, vol. 10, pp. 316–322, 02 2020.
[7] B. Klimt and Y. Yang, "The enron corpus: A new dataset for email classification research," vol. 3201, 09 2004, pp. 217–226.
[8] A. Akinyelu and A. Adewumi, "Classification of phishing email using random forest machine learning technique," *Journal of Applied Mathematics*, vol. 2014, 04 2014.
[9] "Apache spamassassin project," 2006. [Online]. Available: http://spamassassin.apache.org/
[10] J. Nazario, "Phishingcorpus," 2006. [Online]. Available: http://monkey.org/7Ejose/wiki/doku.php?id=PhishingCorpus
[11] N. Unnithan, H. NB, V. R, S. Kp, and S. Sundarakrishna, "Detecting phishing e-mail using machine learning techniques cen-securenlp," 03 2018.
[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[13] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.
[14] A. Ghodsi, "Dimensionality reduction a short tutorial," *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, vol. 37, no. 38, p. 2006, 2006.
[15] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.
[16] I. Rish *et al.*, "An empirical study of the naive bayes classifier," in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.
[17] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010, pp. 45–50.
[18] E. Loper and S. Bird, "Nltk: the natural language toolkit," *arXiv preprint cs/0205028*, 2002.
[19] C. I. Canfield, B. Fischhoff, and A. Davis, "Quantifying phishing susceptibility for detection and behavior decisions," *Human factors*, vol. 58, no. 8, pp. 1158–1172, 2016.
[20] J. S. Downs, M. Holbrook, and L. F. Cranor, "Behavioral response to phishing risk," in *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, 2007, pp. 37–44.
[21] K. Parsons, A. McCormac, M. Pattinson, M. Butavicius, and C. Jerram, "The design of phishing studies: Challenges for researchers," *Computers & Security*, vol. 52, pp. 194–206, 2015.
[22] M. Chatterjee and A.-S. Namin, "Detecting phishing websites through deep reinforcement learning," in *IEEE COMPSAC*, vol. 2, 2019, pp. 227–232.
[23] M. F. Porter *et al.*, "An algorithm for suffix stripping." *Program*, vol. 14, no. 3, pp. 130–137, 1980.
[24] F. Abri, S. Siami-Namini, M. A. Khanghah, F. M. Soltani, and A. S. Namin, "Can machine/deep learning classifiers detect zero-day malware with high accuracy?" in *IEEE Big Data*, 2019, pp. 3252–3259.